



SNAW User Manual

Version: 1.0

Date: May 7, 2020

Team Name: IntelliChirp

Project Sponsors: Colin Quinn and Patrick Burns

Team's Faculty Member: Fabio Santos

Team Members: Steven Enriquez, Michael Ewers, Joshua Kruse, Zhenyu Lei

Introduction	3
Installation	3
Prerequisite 1: Setting Up a Python Virtual Environment with all Needed Python Packages	3
Prerequisite 2: Installing Node JS	4
Setting Up Web Application (Locally)	4
Setting Up Standalone Application	5
Setting Up Soundscape Neural Network	6
Configuration and Daily Operation	6
Web Application (Locally) Daily Operation	6
Standalone Application Daily Operation	7
Soundscape Neural Network - Retraining the CNN Model	7
Maintenance	9
Web Application Trouble-Shooting	9
Conclusion	11

Introduction

We are pleased that you have chosen the Soundscape Noise Analysis Workbench (SNAW) for your business needs. SNAW is a powerful web and standalone application for automatic sound identification from a soundscape using machine learning. This workbench has been custom-designed to meet your needs.

Some of the key highlights include:

- Labelled Spectrograms (Web Application)
- Convolutional Neural Network Analysis on files in the WAV format
- Ability to calculate various acoustic indices on files in the WAV format
- Ability to export the results to CSV

The purpose of this user manual is to help you successfully install and maintain the Soundscape Noise Analysis Workbench going forward. Our aim is to make sure that you are able to easily utilize our product for years to come.

Installation

As part of the final delivery, the Soundscape Noise Analysis Workbench should have been installed on a platform of your choice. Over time, however, you may choose to move to a new platform or re-install the product. Below is a detailed description of how to install the SNAW web application locally on a PC as well as the standalone application.

Before we explain the steps to install the web and standalone application, there are two prerequisites to complete.

Pre-Requisites:

1. [Setting Up a Python Virtual Environment with all Needed Python Packages](#)
2. [Installing Node JS](#)

Prerequisite 1: Setting Up a Python Virtual Environment with all Needed Python Packages

Note: Anaconda is not compatible.

1. Install Python (version 3.7.4) from:
<https://www.python.org/downloads/release/python-374/>
2. Make sure Pip is up-to-date. Run command `py -m pip install --upgrade pip`
3. Navigate to the directory where you would like to install the virtual environment.
4. Run command `py -m venv snaw` (change 'snaw' to what you would like to name the virtual environment).

5. Activate the environment by navigating to the file directory of your environment and running command `.\Scripts\activate`
6. Verify that the virtual environment is running by looking on the left hand side of the bottom line. You should see the name of your virtual environment in parenthesis (Ex: (snaw) C:\.....).
7. Visit SNAW's 'Installation_Requirements' repository at: https://github.com/intelliChirp/Installation_Requirements
8. Click the green 'Clone or download' button.
9. Click on 'Download ZIP'.
10. Copy requirements.txt file to your virtual environment directory.
11. Navigate to your virtual environment's directory.
12. Type the command `pip install -r requirements.txt`
13. Wait for the packages to install on your local machine.
14. Done!

If you ran into trouble during the virtual environment setup, this website may be helpful: [\(https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/\)](https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/)

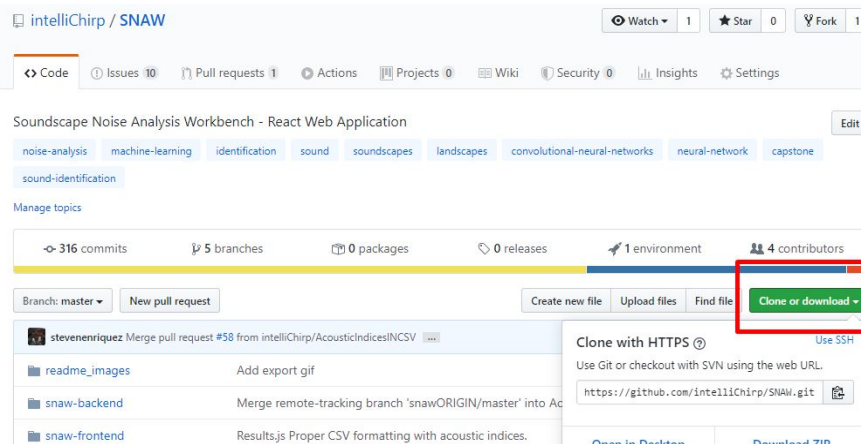
Prerequisite 2: Installing Node JS

1. Visit the website <https://nodejs.org/en/download/>
2. Download the LTS Installer for your Operating System.
3. Run the Installer.
4. Verify that Node JS has installed by opening a terminal and typing `node -v`

Now that the Python virtual environment has been set up and Node JS has been installed, we will now provide detailed steps on how to install the SNAW web and standalone application.

Setting Up Web Application (Locally)

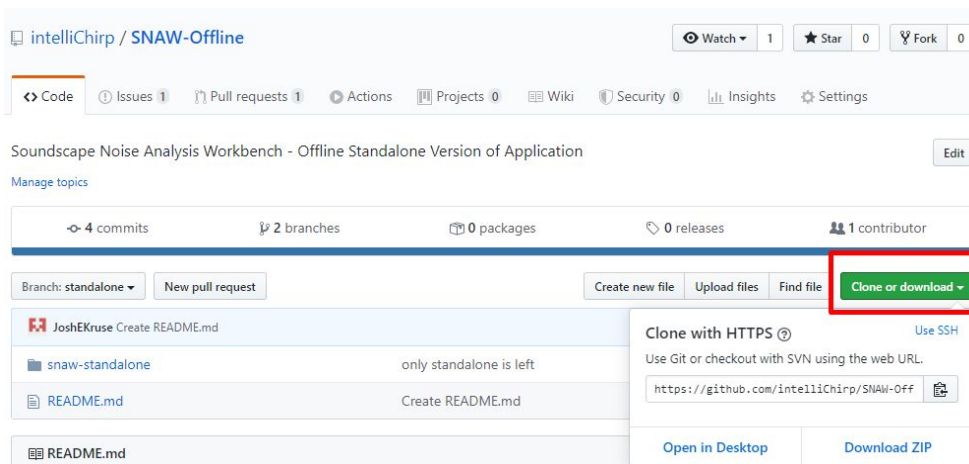
1. Visit the SNAW web application's github page at: <https://github.com/intelliChirp/SNAW>
2. Click on the green 'Clone or download' button



3. Option 1 - If Git is installed on your machine, you can Clone With HTTPS by:
 - a. Copy the link.
 - b. Open a terminal.
 - c. Navigate to a directory where you would like to place this repository.
 - d. Type 'git clone [PASTE LINK HERE]'.
 - e. The repository should now be downloading itself into that file directory.
 Option 2 - Click 'Download ZIP'. Then extract the zip to any file directory.
4. Navigate to the web application's repository on your local machine.
5. Navigate to /SNAW/snaw-frontend.
6. Type the command `npm install`. This will install all needed packages for this application.
 - **Note: To run 'npm' commands, Node JS must be installed on your machine. If you have not installed Node JS at this moment, there is a short guide above titled "Installing Node JS" that will explain how.**
7. Type the command `npm run build`.

Setting Up Standalone Application

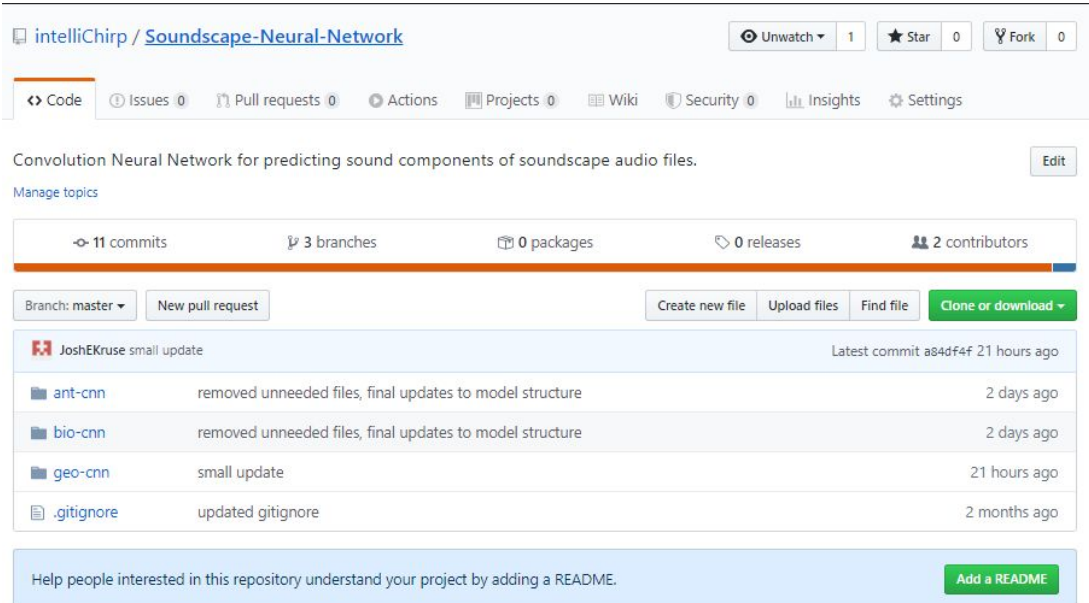
1. Visit the SNAW standalone application's github page at: <https://github.com/intelliChirp/SNAW-Offline>
2. Click on the green 'Clone or download' button



3. Option 1 - If Git is installed on your machine, you can Clone With HTTPS by:
 - a. Copy the links
 - b. Open a terminal
 - c. Navigate to a directory where you would like to place this repository
 - d. Type 'git clone [PASTE LINK HERE]'
 - e. The repository should now be downloading itself into that file directory
 Option 2 - Click 'Download ZIP'. Then extract the zip to any file directory.

Setting Up Soundscape Neural Network

1. Visit the SNAW standalone application's github page at:
<https://github.com/intelliChirp/Soundscape-Neural-Network>
2. Click on the green 'Clone or download' button



3. Option 1 - If Git is installed on your machine, you can Clone With HTTPS by:
 - a. Copy the links
 - b. Open a terminal
 - c. Navigate to a directory where you would like to place this repository
 - d. Type 'git clone [PASTE LINK HERE]'
 - e. The repository should now be downloading itself into that file directoryOption 2 - Click 'Download ZIP'. Then extract the zip to any file directory.

Configuration and Daily Operation

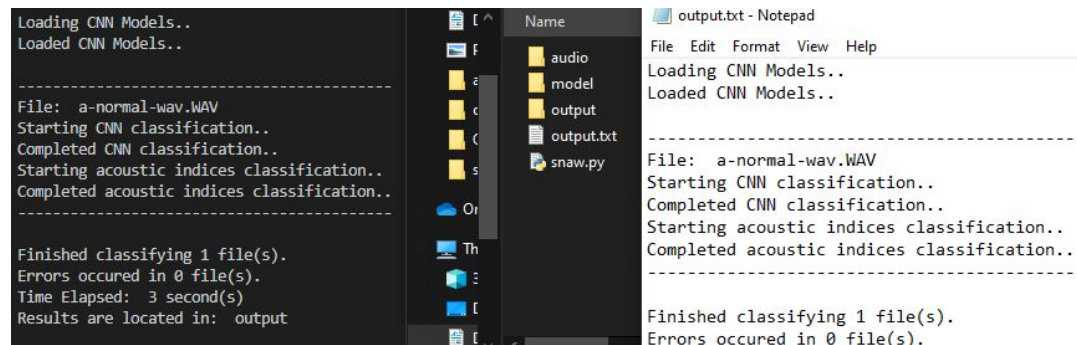
Web Application (Locally) Daily Operation

1. With a terminal open, navigate to the file directory of your Python virtual environment.
Note: If you have not installed a Python virtual environment and the needed Python packages, the section above titled "Setting Up a Python Virtual Environment with all Needed Python Packages" will explain how.
2. Activate the virtual environment by running `.\Scripts\activate`.
3. Navigate to the file directory /SNAW/snaw-backend.
4. Type the command `py api.py`. This will start the server on your machine.
5. Once the command has finished running, the last line printed to the terminal should show: 'Running on [URL-PATH] (Press CTRL+C to quit)'.

6. Open a web browser, paste the url path provided into the URL, and press 'Enter'.

Standalone Application Daily Operation

1. With a terminal open, navigate to the file directory of your Python virtual environment.
Note: If you have not installed a Python virtual environment, the section '**Setup Python Virtual Machine with all Needed Python Packages**' above will explain how.
2. Activate the virtual environment by running `.\Scripts\activate`
3. Navigate to the standalone application's repository on your local machine.
4. Navigate to `/SNAW-Offline/snaw-standalone/`
5. There are already 2 directories created for you to use. Place your WAV files that you would like to classify in the 'audio' directory. The output folder is where the CSV results will be placed after a classification.
6. Type command 'py snaw.py -i audio -o output'
7. After the application completes, your results will be located in the 'output' directory.
Note: The directories do not need to be named 'audio' and 'output'. That is only there by default for convenience. You can create folders with any name, and then use the command format:
'py snaw.py -i [AUDIO-DIRECTORY] -o [OUTPUT-DIRECTORY]'
8. [Windows Command Prompt Users] If you would like to export the output of the application in a text file, add '| tee file-name.txt' to the end of your command. Once the application finishes running, the text file will appear.
 - **Note:** Save this file elsewhere once the classification completes. If you run the application again with the same text file name, the file will be overwritten by the new classification.
 - **Example:** 'py snaw.py -i audio -o output | tee output.txt'



```
Loading CNN Models..
Loaded CNN Models..

-----
File: a-normal-wav.WAV
Starting CNN classification..
Completed CNN classification..
Starting acoustic indices classification..
Completed acoustic indices classification..
-----

Finished classifying 1 file(s).
Errors occurred in 0 file(s).
Time Elapsed: 3 second(s)
Results are located in: output
```

Soundscape Neural Network - Retraining the CNN Model

1. With a terminal open, navigate to the file directory of your Python virtual environment.
Note: If you have not installed a Python virtual environment and the needed Python packages, the section above titled "Setting Up a Python Virtual Environment with all Needed Python Packages" will explain how.
2. Activate the virtual environment by running `.\Scripts\activate`.

3. Navigate to the file directory /Soundscape-Neural-Network.
4. Type the command `jupyter notebook`.
5. You will see three files directories for each CNN (Anthrophony, Biophony, Geophony.)
6. Create a file directory called `/data/`. Place all training files into this folder separated by label. (Each different class needs to be in a seperate folder with the folder name matching the class name)

Name	Date modified	Type
BAM	5/7/2020 11:39 AM	File folder
BBI	5/7/2020 11:39 AM	File folder
BIN	5/7/2020 11:39 AM	File folder
OPI	5/7/2020 11:39 AM	File folder
OQU	5/7/2020 11:39 AM	File folder

7. Open the .ipynb files using the jupyter notebook instance.
8. In the second cell of each notebook you will find a variable called labels. Update these labels with what is featured in your `/data/` file directory.
9. **Adjusting Hyperparameters :**
 - a. In the second cell of each notebook you will find a dictionary titled `hyperparameter_defaults`. This features some of the most important hyperparameters.
 - i. (There are more hyperparameters that you can edit in the layer structure section of the notebook but these are the most important.)
 - b. `max_len` : Sets the length of each training sound clip's MFCC output
 - i. 30 is the value used for 1 second training files
 - c. `buckets` : Sets the amount of MFCC's to return.
 - i. Smaller values lowers the complexity of the training samples
 - d. `epochs` : Sets the amount of full runs through the training set to complete
 - i. When retraining the model set this to a higher value (>20). Then find which epoch returns the smallest validation loss value. Then run the model again with this amount of epochs.
 - e. `batch_size` : Amount of training samples to consider at each epoch
 - f. `layer_one - layer_four` : Hidden unit size at each Neural Network layer
 - g. `dropout_one, dropout_two` : Randomly removes this percentage of training samples at this point in the layer structure
 - h. `sampler` : Type of sampler to use. Useful when classes are unbalanced
 - i. `none` : no sampler will be chose
 - ii. `over` : randomly selects training samples from minority classes and adds them to the list again, until this amount reaches the size of the largest class.
 - iii. `under` : randomly selects training samples from majority classes and removes them until this amount reaches the size of the smallest class

- iv. smote : a random example from the minority class is chosen. The nearest neighbors for that example are found. A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.
10. After adding all the training data and adjusting hyperparameters simply run all cells in the notebook.
11. Output
- a. The model will save as a .h5 file in the file directory. This can be plugged straight into the web application and the standalone application.
 - i. **Web App** : The models are placed in the file directory `/SNAW/snaw-backend/models` in their respective directory
 - ii. **Standalone** : The models are placed in the file directory `/SNAW-Offline/snaw-standalone/models` in their respective directory
 - b. A summary of the model structure will appear.
 - c. Accuracies for every class and the overall model will appear.
 - d. A Confusion Matrix will be created and can be downloaded.
 - e. A table of precision, recall, f1-score, accuracy, macro accuracy, weighted accuracy, and zero one loss will be calculated.
 - f. ROC curve graph will be calculated for every class.

Maintenance

For the web application, maintenance is taken care of for you. When a user leaves the application, his/her user folder should be deleted as well as all of the files that were uploaded. Over time, it will be best practice to check up on that directory to see if any user directories were not deleted.

To accomplish this, navigate to ‘\snaw-backend\instance\upload’. Delete any user directories in this directory. You can safely delete everything in the upload directory.

Web Application Trouble-Shooting

Errors may occur on a WAV file that shouldn't happen. Whether the application cannot create the spectrogram and run analysis, or the acoustic indices are not generated. These specific occurrences can be found within the code base for testing, if a file seems to have issues.

Spectrogram/Analysis Failure

- If the application cannot create a spectrogram for a WAV file, then the analysis will also not complete. The creation of the Spectrograms is found in the `get_spectrogram.py` file, located in the method signature:
 - ```
def runScript(filename, fileCount, audiofile, data):
```

- The spectrogram images are created in the following lines in the method signature above:

```
encode_ant, wavEncode = encoding(data[0]['data'], audiofile, path)
encode_bio, _ = encoding(data[1]['data'], audiofile, path)
encode_geo, _ = encoding(data[2]['data'], audiofile, path)
encode_none, _ = encoding(None, audiofile, path)
```

These lines are included within a “Try/Except” block. Which when fired will print any error/(s) to the terminal. any errors may be solvable through further research on the error code provided.

- The **Except** block of code attached to the **Try** which houses the lines above is here:

```
except:
 #if DEBUG_FLAG : print('[FAILURE -- Spectrogram] File upload
 unsuccessful, or no file uploaded.')
 track = traceback.format_exc()
 print(track)
 image_list = ERROR
 audio_wav = ERROR
```

### Acoustic Indices Failure

- If the acoustic indices of the file cannot be calculated, then the web application will not show the results of the neural network running. The acoustic indices creation can be found in the file **acousticIndices.py** located in the method signature:

- **def getAcousticIndices(audiofile):**

- The acoustic indices are created in the following lines in the method signature above:

```
acousticIndices = AcousticIndices(data_chunk,new_fs)
acoustic_indices = acousticIndices.get_acoustic_indices()
acoustic_indices = list(map(lambda x: round(x, 4), acoustic_indices))
```

- The **Except** block of code attached to the **Try** which houses the lines above is here:

```
except Exception as e:
 track = traceback.format_exc()
 print(track)
 singleResultArray = "ERROR_PRESENT"
```

These lines are included within a “Try/Except” block. Which when fired will print any error/(s) to the terminal. any errors may be solvable through further research on the error code provided.

## Conclusion

We wish you many years of classifying with the Soundscape Noise Analysis Workbench. We are very glad to be able to help with this solution. In the coming months, we will be available by email to answer any questions that arise.

### **Best wishes from your Soundscape Noise Analysis Workbench developers**

Steven Enriquez ([stevenenriquez123@gmail.com](mailto:stevenenriquez123@gmail.com)),  
Josh Kruse ([heyjoshkruse@gmail.com](mailto:heyjoshkruse@gmail.com), [jek248@nau.edu](mailto:jek248@nau.edu)),  
Michael Ewers ([ewers.sw@gmail.com](mailto:ewers.sw@gmail.com)),  
Zhenyu Lei



**IntelliChirp**